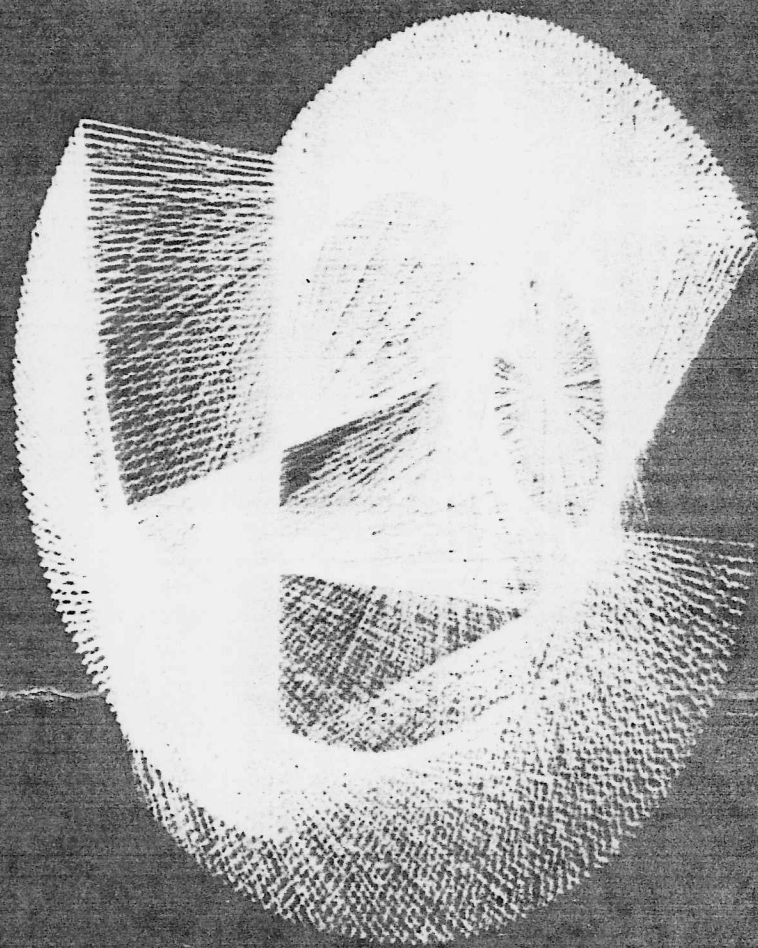


CG24102

Massachusetts
Institute
of Technology

Project MAC
Progress Report III
July 1965 to
July 1966



CG24102

work, for example, on list-structure programs which do not display any particular page-use patterns. Detection of Working Sets will work for any process which does not change working-set size too rapidly with respect to the sampling interval. Inasmuch as the optimum page-removal decision used information regarding processes in the Ready List, an intimate relationship between Scheduling and Allocation has been uncovered. (See Denning, Appendix B; MAC-TR-21, Appendix D; and Project MAC Computation Structures Group Memo No. 24, March 1966.)

Digital Logic Simulation* - Donald L. Smith

A data structure has been developed for describing complex digital systems for purposes of simulation, design evolution, and logic verification. The principle criteria used in formulating the data structure were: 1) speed of simulation for large systems having asynchronous timing, 2) provision for incremental design changes without regeneration of the entire data structure, 3) ability to fully detect timing hazards if this is desired by the designer.

The basic elements of the data structure represent flip-flops, combinational logic blocks, delays, and events. These elements are tied together in a list structure that facilitates alterations and makes efficient simulation possible. The simulation algorithm queues future events according to their time of occurrence, and evaluates a combinational level only when its value is needed and may have changed since last evaluated.

The proposed data structure is viewed as the basis for a digital logic design system that would be implemented on a time-shared computer, and would include a conversational design translator as well as the simulator and logic checking aids.

* See Smith, Donald L., Models and Data Structures for Digital Logic Simulation, M. I. T. Department of Electrical Engineering, M. S. Thesis, June 1966 (soon to be a Project MAC Technical Report).

Development of the PDP-1-X Multiuser System - Jack B. Dennis and Leo J. Rotenberg

Experience obtained operating the PDP-1 time-sharing system, at the M. I. T. Research Laboratory of Electronics, together with experience at Project MAC, has taught us much about desirable properties of time-shared computer systems, or, in general terms, multiprogrammed computer systems. Features of a Multiprogrammed Computer System, (MCS) are generally evident in the services that an object program may obtain from the system by giving commands that we call meta-instructions. From work done at Project MAC, Dennis and Van Horn have published a set of meta-instructions and defined their meaning in terms of a program structure felt to be particularly suitable for an MCS. Their principal ideas concern parallel processing, relations among computations, and the organization and manipulation of directories of files, etc., retained by the system on behalf of its users. (See MAC-TR-23, Appendix D.)

With the purpose of providing a realization of many of these ideas, a new multiprogramming executive system has been designed around the Digital Equipment Corporation PDP-1 computer, donated to the Department of Electrical Engineering in 1961. The new system involves hardware additions and modifications, as well as preparation of a new executive program. The modified machine will be called the PDP-1-X. The more important services provided by this system are:

1. File Memory

The PDP-1-X will include a file memory system capable of storing information on miniature tape units (DECtape), so that a user may retrieve programs and data by command from his console. Consistent with the informal way in which this system is operated, a user will ordinarily bring a tape containing his programs with him and mount it on a free unit at the beginning of a session of use, and take an updated tape with him when he leaves.

2. Parallel programming

We have found the notion of process to be more fundamental to the design of an MCS than the concept of program. A process is the sequential execution of the steps of a procedure. One process may, by means of a fork meta-instruction create a second process, which is thought of as running concurrently with the first. The use of a join meta-instruction allows a process to test whether two or more asynchronous activities have been completed. Execution of a quit meta-instruction terminates a process. The fork, join, and quit meta-instructions form one set of primitive operations for parallel programming. In the PDP-1-X, use of these primitives to control concurrent input/output activity replaces the interrupt feature that serves this function on most present machines.

3. Computations

We use the term computation to mean a group of processes that collectively accomplish a computing task. Each computation runs on behalf of a user of the system and has an associated sphere of protection that determines what files, services, or directories the processes of the computation are allowed to access or use. In our revised system, the sphere of protection is established by a 64-entry program reference list through which the processes of the computation execute input/output operations, gain access to directories and files, and obtain other services provided by the executive system. A primary computation will be able to run a second computation under its surveillance. This is a valuable asset in program debugging; it allows a programming system running as an object computation to monitor a second object computation under test, without the danger of being disabled if the test process should run wild or address incorrectly.

4. Directory Structure

The system design includes a hierarchical directory structure. Each item in a directory associates an alphanumeric name with a pointer to a file, an input/output function, an entry point giving access to some system service, or a directory giving further associations of names with pointers. Processes may reference items in a subtree of the directory lattice by specifying a downward path from a directory that is accessible through the capability list of the computation. A system user's root directory is always accessible through the capability list of a computation operating on his behalf. By means of meta-instructions (similar to those defined in MAC-TR-23), processes will be able to share files in a rather flexible way.

The most important innovation of this system is the provision of environment in which parallel programming may be used and developed. Most functions of the executive program are performed by executive processes, scheduled without preferential treatment, that run concurrently with user processes and use the parallel programming facilities for control of input/output functions.

Because of our emphasis and reliance on parallel programming, it is essential that the use of these features involve minimal cost in processing time. For this reason, specially designed process-scheduling hardware is being built into the PDP-1-X, so that a particularly fast implementation of the fork and quit primitives is obtained. There is little bookkeeping to be done by the executive when switching control between processes, and we anticipate switching times in the range of 200 to 300 microseconds. (See also, L. J. Rotenberg, Proposal: An Implementation of an Almost-Segmented Multiprogrammed Computer System for the PDP-1, Project MAC Computation Structures Group Memo No. 19.)

Academic Staff

F. J. Corbató
A. Evans, Jr.

E. L. Glaser
J. W. Poduska

J. H. Saltzer
C. Strachey

Non-Academic Research Staff

M. J. Bailey
M. E. Child
G. Clancy
P. A. Crisman
C. A. Cushing
R. C. Daley
S. D. Dunten
D. J. Edwards
C. Garman

R. M. Graham
H. C. Haig
B. W. Kernighan
C. Marceau
N. I. Morris
M. Padlipsky
E. Quisenberry
R. Rappaport
S. L. Rosenbaum

G. G. Schroeder
P. E. Smith
M. K. Thompson
T. H. Van Vleck
L. Varian
D. Wagner
D. R. Widrig, Jr.

Research Assistants and other Students

H. Magnuski

B. D. Wessler

Development of the MULTICS system is being carried out with the cooperation of the research staff of the Bell Telephone Laboratories and the General Electric Company.

The Transition from CTSS to MULTICS - F. J. Corbató

A. CTSS MAINTENANCE

During the reporting period, the group's primary effort has been shifted from maintaining the Compatible Time-Sharing System (CTSS) to developing the MULTiplexed Information and Computing Service (MULTICS). The final major activity on the development of CTSS was completion of a new file system which has allowed increased flexibility; enabling users to simultaneously read files as well as "link" to each others files. The file system was finally installed during August and September of 1965. The file system was difficult to install, not only because of the considerable magnitude of programming required, but also because almost all commands had to be modified. In addition, it was necessary to write special programs for dumping and reloading the disc, and salvaging the disc in the event of system mishaps (i. e., making the disk syntax correct); requesting the retrieval of files which had been removed from the disc; and programming for the standard input and output of information to and from the disc file. The final step required to make the new file system complete was addition of the DAEMON program for incrementally dumping a user's newly created files whenever he has logged out. This system, which is now completely installed, obviates the periodic total dump which was formerly required. Furthermore, the DAEMON should prevent file losses from being catastrophic when a system mishaps occurs, inasmuch as the backup is more closely tailored to user behavior than was the old system.

As would be expected with an evolving and maturing system, there are numerous new commands, and improvements have been made to old commands. Some of these changes are for improved administration of the system, such as upgrading of the LOGIN command, and proper record keeping of the console identification of each user, the system identification, and each user's last LOGOUT time. Another added administrative command is TTPEEK, which allows a user to inspect both the allotments and usage of his central processor time, as well as his disc, drum, and tape records.

Other changes have included the introduction of a MAIL command, which allows users to send notes to other users, on-line, to be deposited in individual "mailboxes" within each user's file directory. This feature has rapidly become a standard communication technique within the system. A further change in the command system has been the production of special commands to produce smooth operation of the RUNCOM command, which allows sequences of commands to be treated as macro-commands. RUNCOM has turned out to be one of the most powerful additions to the CTSS system, since it essentially allows users to arbitrarily create new commands of great complexity.

Inclusion of magnetic tape operation from foreground consoles within the time-sharing system has been another important step within the past period. Tape usage has turned out to be a valuable service, although out of necessity it must remain limited as there are not that many tape devices compared to users of the system. This feature has been a key factor in providing remote access to the simulation system for the GE 645.

A long-awaited feature of the time-sharing system has been the ability to leave Foreground-Initiated Background (FIB) jobs to be run in the absence of the user. In other words, the benefits of batch processing are flexibly brought to a time-sharing user such that he is able to disassociate himself from operating a well-behaved program. The FIB system has not only been useful as an extension of the system for time-sharing users, but has been of great assistance in implementing various automatic-accounting procedures; wherein programs "go to sleep" for periodic times and are subsequently reawakened to do various administration chores during intervals varying from minutes to weeks.

Additionally, there are restricted classes of users who are only able to utilize a subset of commands, and various changes and modifications have been made to improve and tighten system security against both accidents and vandalism.

B. CTSS DOCUMENTATION

In the area of documentation, one problem has been that as system size and complexity has grown, responsibility for maintenance has become more diffusely distributed: it has been more and more important to develop a systematic approach to maintenance. Shifting the maintenance staff away from CTSS was a fitting opportunity to draw up an inventory of the system components. This inventory consisted of an itemization of the modules in: 1) the main supervisor, 2) the command system, 3) the library system of subroutines, and 4) the auxiliary programs required to continue to operate the system (e. g., the Salvager, etc.). This inventory of the system has allowed responsibility to be sensibly delegated, and has been valuable as a check list when changes to the system have been contemplated.

The second edition of the *CTSS Programmer's Guide* was published at the beginning of this period in looseleaf form for easy maintenance. Furthermore, an up-to-date version of the manual is maintained on-line in CTSS using the TYPSET program, so that users may examine this copy from their consoles for last-minute changes. (See Crisman, Appendix C.)

C. MULTICS DESIGN PHILOSOPHIES

The primary efforts of the Multics workers were aimed at determination of final system hardware specifications, and the design and development of system software. Inherent in both of these efforts was the underlying requirement that all work on the project be thoroughly documented. Thus, a large body of reports and manual sections have been built up during the past year.

Multics is a joint effort of Project MAC, Bell Telephone Laboratories, and the General Electric Company. Its basic design philosophies were formally presented in a series of six papers, by representatives of all three organizations, at the 1965 Fall Joint Computer Conference. In their paper, "Introduction and Overview of the Multics System", F. J. Corbató of M.I.T., and V. A. Vyssotsky of BTL described important features of the new system:

"The overall design goal of the Multics system is to create a computing system which is capable of comprehensively meeting almost all of the present and near future requirements of a large-computer-service installation. It is not expected that the initial system, although useful, will reach the objective; rather, the system will evolve with time in a general framework which permits continual growth to meet unknown future requirements. Use of the PL/I language will allow major system software changes to be developed on a schedule separate from that of hardware changes ... It is expected that the Multics system will be ... available for implementation on any equipment with suitable characteristics ..."

Some of the important hardware features of the Multics system were also reported in this paper. They include: two-dimensional addressing (segments); "paged" memory; execute-only segments, which have particularly important applications for service bureaus and other applications where a program is to be executed but not read; a General Input/Output Controller (GIOC) which removes much of the input/output responsibility from the central processor, and, at the same time, makes the use of different input/output devices more flexible from the point-of-view of the programmer; and a full 128-character ASCII code. Another major hardware feature inherent in the basic design of the Multics system is the high level of system reliability that may be achieved. This is partly due to the fact that, because there will generally be more than one of each major component, a failure in any one component will not disable the entire system. This feature will also provide for more flexible maintenance, since the system can be partitioned into two distinct systems.

In addition to the paper already mentioned, five other papers presented at the conference were: "System Design of a Computer for Time-Sharing Applications", by E. L. Glaser of M.I.T., and J. F. Couleur and G. A. Oliver, both of G.E.; "Structure of the Multics Supervisor", by V. A. Vyssotsky of BTL, and F. J. Corbató and R. M. Graham, both of M.I.T.; "A General Purpose File System for Secondary Storage", by R. C. Daley of M.I.T., and P. G. Neuman of BTL; "Communications and Input/Output Switching in a Multiplexed Computing System", by J. F. Ossanna, L. E. Mikus, and S. D. Dunten, of BTL, G. E., and M.I.T., respectively;

and "Some Thoughts about the Social Implications of Accessible Computing", by E. E. David, Jr., of BTL, and R. M. Fano of M.I.T. (See Proceedings of the 1965 Fall Joint Computer Conference; Spartan Press; Washington, D.C.; pp. 185-247.)

D. MULTICS SOFTWARE DEVELOPMENT

Development of the software system has centered in four major areas, the first of which was development of a first design for the File System which is used to store segments. This generalized version of the file system used in CTSS introduces a hierarchical file structure which permits referencing to any number of levels. The system of links and permissions has been made more general, with linkage permission being associated directly with a permitted file. In addition, the file system incorporates such additional considerations as selective levels of user privacy, ease of data movement, and a backup mechanism which is readily-available in the event of either user error or system malfunction.

The design for the Central Traffic Controller was documented in a Ph.D. dissertation by J. H. Saltzer. (See Saltzer, Appendix B.) In his abstract, the author briefly describes his scheme for handling many users simultaneously:

"The scheme is based upon a distributed supervisor which may be different for different users. The processor multiplexing method provides smooth inter-process communication, treatment of input/output control as a special case of inter-process communication, and provision for a user to specify parallel processing or simultaneous input/output without interrupt logic. By treatment of processors in an anonymous pool, smooth and automatic scaling of system capacity is obtained as more processors and more users are added. The basic design has intrinsic overhead in processor time and memory space which remains proportional to the amount of useful work the system does under extremes of system scaling and loading. The design is not limited to a specific hardware implementation; it is intended to have wide application to multiplexed, multiple-processor computer systems."

A third area of development involved I/O software for the previously mentioned General Input/Output Controller. The I/O system incorporates

such features as stream switching, where the internal program may accept input and generate output without concern for the specific I/O device being used (indeed, these devices may be switched during the execution of a process); a standard character set, with both standard internal codes, external graphics, and a standard escape character, all irrespective of the particular I/O device being used; and a canonical string, where all strings of characters that look alike are represented alike within the system (this concept has been partially implemented within the context editor TYPSET in CTSS, but has now been made a standard part of the Multics I/O system).

The fourth key area of effort has been directed toward development of a command system. This includes a functional command language, similar in philosophy to CTSS. One feature of the command language is that many of the elaborate details need not be learned by the user if they are not needed in his particular work. In structure, the command-language interpreter is a system for calling upon a large library of subroutines to actually execute individual commands. Because of this, commands may be called and executed at any level; from a user console or within a running program.

As mentioned earlier, thorough documentation is an important part of the research effort in the Multics project. Consequently, details of the command language and other components of the system have been assembled in the Multics System Programmer's Manual. This document is being constantly changed, updated, and added to. A basic groundrule is the standard engineering practice that all software subsystems must be thoroughly designed, and implementation plans specified, before actual coding begins. We expect that this procedure will result in better integration and more effective operation of the resultant system than would otherwise occur.

E. MULTICS IMPLEMENTATION

To implement and debug the Multics system in a reasonable amount of time after it has been designed, a parallel tool-building effort has been carried out by various members of the three organizations involved. A GE 635

was delivered in July 1965 and was operating one month later. The following tools have been implemented:

1. A compiler for a subset of PL/I (called EPL for "Early PL") in CTSS;
2. An editing and input program (EDA) in CTSS using the Multics standard 128 ASCII characters which are typed in using appropriate escape conventions;
3. A program (merge-edit) in CTSS which writes a magnetic tape on the IBM 7094, in a format which is acceptable as an input tape for the GE 635 monitor (GECOS);
4. An assembly program for the GE 645 which runs in GECOS;
5. A program which simulates the GE 645 and runs in GECOS;
6. A program running in GECOS which loads a Multics process and starts simulation;
7. A collection of procedures (written in GE 645 assembly language) which provide the minimum services necessary to execute a Multics process;
8. A program to write on magnetic tape, the simulated GE 645 memory (when the simulated process terminates), and the results of any assemblies, in a format suitable for the CTSS disk editor;
9. An on-line, symbolic, interactive program in CTSS (GEBUG) which is used to examine the contents of the simulated GE 645 memory after it has been returned from the GE 635.

This complex of tools allows a system programmer to input, edit, compile, assemble, execute, and examine the results of a Multics process from any remote console attached to CTSS.

First units of the GE 645 were scheduled for September 1966 delivery. The simulation and debugging system just described will be used on the GE 645 until such time as the first version of Multics can stand by itself. It is expected that this will be in mid-1967.

APPENDIX A

PROJECT MAC MEMORANDA

MEMORANDUM MAC-M-No.	SUBJECT	AUTHOR	DATE
258	Design of the Hand (AIP memo 86)	M. L. Minsky	8/13/65
259	AED Flash 19: RWORD Package (9442-M-141)	R. J. Bigelow	8/12/65
260	AEDBUG Program Description (9442-M-142)	T. B. Fox	8/16/65
261	Modifications and Addenda to the AED-O Programmer's Guide (9442-M-143)	C. Feldmann, C. Bower, P. Ladd	8/18/65
261-2	Additional Modifications and Addenda to the AED-O Programmer's Guide (9442-M-143-2) [Does not replace MAC-M-261]	C. Feldmann	9/7/65
262	A Theory of Computer Instructions (AIP memo 89)	W. D. Maurer	9/65
263	FLIP - A Format List Processor (AIP memo 87)	W. Teitelman	9/16/65 (not dated)
264	EDIT and BREAK Functions for LISP (AIP memo 88)	W. Teitelman	9/20/65 (not dated)
265-1	Changes to Teletype Terminals (CC-265)	R. G. Mills	11/23/65
266	Relations Between the Project MAC System and a User	A. I. Dumey	9/13/65
267	MACTAP, a PDP-6 DEctape Handling Package (AIP memo 87-A)	P. Samson	9/16/65
268	Linking Loader for MIDAS	P. Samson	1/31/66
269	Number Conversion Subroutine CONVERT (944-M-157)	J. V. Oldfield	11/19/65
270	MADIO, a Simplified Input-Output Package for MAD Programs	P. J. Denning	9/65
271	AED Flash 20: New Array Handling Language for AED-O (9442-M-145)	P. Ladd	9/24/65
272	AED Flash 21: Octal Debugging Subroutine LOPAT (9442-M-146)	J. Walsh	9/24/65
273	The ODA String Package (9442-M-150)	D. T. Ross	10/1/65

MEMORANDUM MAC-M-No.	SUBJECT	AUTHOR	DATE
274	AED Flash 22: Use of Remote Display Consoles (9442-M-148)	J. Rodriguez	10/7/65
275	AED Flash 23: Use of GENCAL (9442-M-152)	J. Walsh	10/14/65
276	Direct-View Storage-Tube Displays	R. H. Stotz, J. E. Ward	3/8/66
277	OPS-3 Goes Public (replaces MAC-M-233)	M. Greenberger, M. M. Jones, D. N. Ness, J. H. Morris	10/25/65
278	AED Bibliography (9442-M-153)	D. T. Ross	10/20/65
279	MIDAS (PDP-6 assembly program) (AIP No. 90)	P. Samson	10/65
280	Operating Statistics of the MAC Time-Sharing System	T. Hastings	12/8/65
281	Computer-Assisted Instruction (CAI)	R. Rosenberg	10/20/65
282	Computer Experiments in Finite Algebra - II	W. D. Maurer	12/1/65
283	AED Flash 26: BSS Plex Dump (9442-M-156)	R. O. Ladson	11/2/65
284	Usage Conventions for EPS SAVED	C. C. Tillman	11/30/65
285	A Useful Algebraic Property of Robinson's Unification Algorithm (AIP No. 91)	T. P. Hart	11/65
286	AED Flash 25: The Loader/Unloader (9442-M-155)	B. L. Wolman	11/16/65
287	A Display Buffer Computer for the ESL Console	J. Grondstra, R. H. Stotz, J. E. Ward	11/23/65
288	An Experiment Using a New System of Documents for Communication in the Computer Science Field	R. M. Fano	12/28/65
289	Operation Notes for the DAEMON (CC-258)	G. Clancy	12/30/65
290	Complete CTSS Inventory (CC-259) [Limited Distribution]	D. R. Widrig, G. Schroeder, M. M. Child	1/3/66

MEMORANDUM MAC-M-No.	SUBJECT	AUTHOR	DATE
291	AED Flash 27: New CTEST2 Command (9442-M-158)	C. Feldmann	1/7/66
292	Some Remarks on the Problem of Exponential Growth in Machine Theorem Proving	W. D. Maurer	1/7/66
293	Availability of the MAP System (CC-260)	J. W. Brackett	2/3/66
294	Topics in Model Theory (AIP No. 92)	M. I. Levin	1/20/66
295	The Value of Compatibility Between Time-Sharing and Batch Processing	W. D. Maurer	1/20/66
296	A New Version of CTSS LISP (AIP No. 93)	R. Fenichel, J. Moses	2/2/66
297	The Appropriateness of a Computer Programming Department As Opposed to a Computer Sciences Department	W. D. Maurer	1/20/66
298	Analysis of a Protected Service Routine (MSG Memo 21)	P. J. Denning	2/21/66
299	A New Machine Learning Technique Applied to Checkers (AIP No. 94)	A. D. Griffith	3/9/66
300	Graphical Input-Output Techniques in Power System Computation (9442-M-159)	J. V. Oldfield	2/28/66
301	Analysis and Transformation of Computational Processes (9442-M-162; MSG Memo 22)	J. Rodriguez	3/7/66
302	Notes on Searching and Sorting	J. W. Poduska	3/1/66
303	MLPL: A MAD-compiled List Processing Language	J. W. Poduska	2/10/66
304	Operators for Manipulating Language Structures (9442-M-160)	B. L. Wolman	3/18/66
305	A Program Feature for CONVERT (AIP No. 95)	A. Guzmán H. McIntosh	4/66
306	Operating Procedures for CTSS (CC-261)	L. P. Odland	4/25/66
307	CTSS SNOBOL User's Manual (CC-235-4)	D. Shea	5/16/66

MEMORANDUM MAC-M-No.	SUBJECT	AUTHOR	DATE
308	POLYBRICK: Adventures in the Domain of Parallepipeds (AIP No. 96)	A. Guzmán	5/66
309	AED Flash 28: Reduction of Load Time (9442-M-168)	B. L. Wolman	5/31/66
310	Symbolic Integration (AIP No. 97)	J. Moses	6/10/66
311	AED Flash 29: Description of "LAED Command" (9442-M-169)	B. L. Wolman	6/10/66
312	General Description of the AED-1 Processor and the Display Interface System (9442-M-170)	D. T. Ross	6/16/66
313	PDP-6 LISP (AIP No. 98)	P. Samson	6/16/66
315	AED Flash 30: Writing "Machine-Independent" AED-O Programs (9442-M-171)	J. Rodriguez	6/21/66
316	CONVERT (AIP No. 99)	A. Guzmán H. McIntosh	6/66
. . . REVISIONS. . .			
192-2	Unrecognizable Sets of Numbers (AIP Memo 73 revised)	M. L. Minsky S. A. Papert	9/15/65
224-1	Subroutines for Automatic Plotting	T. Stockham J. W. Brackett	2/23/66
231-2	Description of SNOBOL Commands (CC-235-3)	L. Pouzin	10/1/65
265-1	Changes to Teletype Terminals	R. G. Mills	11/23/65
278-1	AED Bibliography (9442-M-153-1)	D. T. Ross	3/21/66
289-1	Operations Notes for the DAEMON (CC 258-1)	G. Clancy	5/31/66

OF MANAGEMENT

Security Classification